Towards Dependable Number Entry for Medical Devices

Abigail Cauchi[1] Paul Curzon[2] Parisa Eslambolchilar[1] Andy Gimblett*[1]
Huayi Huang[2] Paul Lee[3] Yunqiu Li[1] Paolo Masci[2] Patrick Oladimeji[1]
Rimvydas Rukšėnas[2] Harold Thimbleby[1]

(1) Swansea University, (2) Queen Mary University of London, (3) Singleton Hospital, Swansea

CHI-MED: Computer-Human Interaction for Medical Devices
www.chi-med.ac.uk

ABSTRACT

Number entry is an ubiquitous task in medical devices, but is implemented in many different ways, from decimal keypads to seemingly simple up/down buttons. Operator manuals often do not give clear and complete explanations, and all approaches have subtle variations, with details varying from device to device. This paper explores the design issues, critiques designs, and shows that methods have advantages and disadvantages, particularly in terms of undetected error rates.

Author Keywords

Medical devices; modelling; formal methods; HCI; healthcare; number entry

Note. *This is a working paper that we will develop further through interactive workshop participation. We will engage additional authors as necessary for continued work to progress towards a high-quality journal paper fully covering the relationship of all relevant medical, manufacturing and computing factors. It is an important topic that we want to get right.*

1. INTRODUCTION

There are many applications where numbers have to be entered into computer systems, from setting alarm clocks to programming infusion pumps. In most applications the consequences of mistakes are limited, but in many cases—in particular with medical devices—they are potentially critical. Mistakes in entering numbers into infusion and syringe pumps could lead to incorrect doses being delivered, causing harm.

There are several inter-related properties of importance in a safety-critical number entry system: efficiency in entering numbers, the likelihood that errors are made and the efficiency of recovery from error [1]. In a hospital it is vital that nurses can use pumps efficiently as they are very busy and multitasking is the norm. Observational studies have suggested that nurses may frequently make minor mistakes in entering numbers, for example not following the 'golden path' that is the most efficient way of entering a particular number, but that these errors are caught and corrected. Thus it might be argued that number entry is not a particularly severe safety critical problem; however, efficiency remains an important concern in a busy ward. Therefore a device where such mistakes do not need to be constantly corrected or where the golden path is most often the one naturally followed would provide significant benefit, given the number of times such devices need to be set. Furthermore, work in resilience engineering suggests that single mistakes rarely lead to disasters. It is when a range of different causes combine. If a large number of trivial and normally unproblematic errors are being made then this increases the potential for other rarer causes to interact with them and lead to a critical incident—as in Reason's "swiss cheese model" [5].

If a patient is given an incorrect drug dose, perhaps ten times higher than intended, the patient may die or have some other adverse outcome. It is therefore crucial that number entry is dependable, that there are no design defects, no mismatches between user conceptual models and device behaviour, and that users can (so far as reasonably possible) detect and correct their errors. This paper shows that this problem is more intricate than might appear at first sight, that many medical devices and their operator manuals fall short, and that better solutions are possible.

*Corresponding author.

Our goal is to identify a set of properties that programmers of medical devices should implement—or if we cannot do that, to recommend a set of key properties to consider before implementation—to minimize error rates, specifically for number entry. It is not obvious how to do this, as it involves a variety of tradeoffs, and thus we propose a debate within the EICS4Med workshop to explore the issues. We bring to the debate prepared material and a variety of demonstration resources to explore ideas. In this paper we highlight the issues involved to promote that debate.

1.1 Typographic conventions

We render arrow keys pressed by users as: ◄ ► ▲ ▼. We represent number displays with a box around each visible digit, some of which might be empty. For example, | 2 | 0 | 9 | . | 4 | shows a six-digit display with two decimal places, showing the number 209.4, with the cursor in the tens column; if the display were reduced to only one decimal place, we'd write it as | 2 | 0 | 9 | . | 4 |

2. PRIOR WORK

There is much prior work on user interface design principles in general, such as Nielsen's Usability Engineering [4], but they are very vague for programmers. For example, undo (which Nielsen recommends) can be implemented in many ways.

Work on human computer interaction specifically linked to number entry is varied and little has been applied specifically in the medical domain.

For example, Hourizi and Johnson [2] consider a number entry error that resulted from a mode error, and which led to the crash of an A320 airliner with loss of life. They argue that this should not be seen as a perception or knowledge error, but rather as due to an inadequate communication protocol between pilot and autopilot; a variation on the design based on this hypothesis was found to eliminate the error in simple user tests.

Brumby et al. [1] investigated trade-offs between efficiency of entering mobile phone numbers vs avoiding errors in driving. Their analysis suggests that interleaving number entry at chunk boundaries efficiently trades the time given up to dialling with that of ensuring enough attention is paid to driving to avoid drifting.

It is well known that device design can encourage certain number entry errors in medicine. For example Zhang et al [7] report an incident where a nurse intending to program a pump at 130.1ml/h inadvertently programmed the pump at 1301ml/h — a rate 10 times larger than the intended rate. Unknown to the nurse, the decimal point on the interface of the pump only works for numbers up to 99.9.

Thimbleby and Cairns [6] show that out by 10 errors in number entry systems, like the one described above, can be halved with better interaction design focussing on error management.

Figure 1. Screenshot of interactive Alaris GP simulation

3. EXAMPLE DEVICES

We have investigated and simulated a number of medical devices in order to explore their behaviour and related HCI issues; in this section we introduce the two particular devices, both infusion pumps, whose number entry behaviour is both typical and interesting, and around which the rest of this paper is built.

The Alaris GP infusion pump (figure 1) exemplifies a number entry interface style found on a variety of syringe and infusion pumps: two pairs of buttons change the displayed value; one pair increases the value, the other decreases it. In each pair, one of the buttons causes a bigger change than the other. Each button can also be held down to increase the rate of change of the number on the screen.

The B.Braun Infusomat Space pump (figure 2) has three distinct number entry systems used for different tasks, all based around a set of ◄ ► ▲ ▼ buttons; it exhibits a number of interesting behaviours. It is a good example of the way in which number entry is widely perceived as unproblematic and trivial, while in fact harbouring potential for surprises and difficult. Its user manual has very little to say on the topic: "When editing parameters, switch digits/levels using ◄ ►. White background indicates current digit/level. Use ▲ or ▼ to change current setting." Elsewhere in the manual, the arrows are described as: "Arrow up and down: Scroll though menus, change setting of numbers from 0-9, answer Yes/No questions. Arrow left and right: Select data from a scale and switch between digits when numbers are entered. Open a function while pump is running or stopped with the left arrow key."

This description is inadequate; for example, it suggests

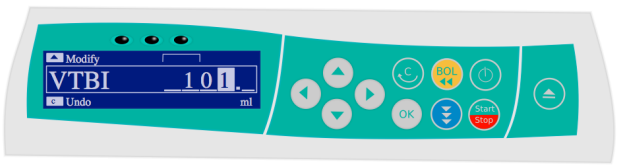

Figure 2. Screenshot of interactive B.Braun simulation

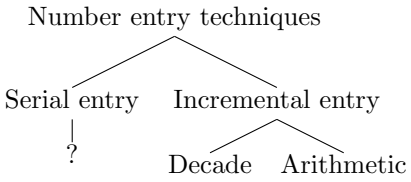

Figure 3. Number entry—basic classification

that if the display is 9 (say) and ▲ is pressed, then the display will become 0. In fact, it becomes 10, i.e. an arithmetic operation was performed (9 + 1).

More concerningly, if the display is 10 and ▼ is pressed, it becomes 0.001. The arithmetic operation performed in this case was 10 - 100 = -10, which result was then clamped to a minimum value, 0.1. It is easy to imagine scenarios in which this behaviour leads to an underdose, perhaps harmfully. The pump has similarly surprising and inconsistent behaviour around the maximum value. These issues are described further in the next section.

For a new user, the infusion pump is likely to behave unpredictably, though we do not know what implications this unpredictability has on safety in medical scenarios. The lack of symmetry between minimum and maximum behaviour might have an impact on usability, as do the arithmetic operations, particularly when subtracting a value which results in a number less than 0.

4. RESOURCES FOR DEBATE

In order to support debate around these issues, we will bring a range of resources to the workshop.

Simulations — We have implemented a variety of user interfaces for entering numbers, closely based on real infusion pumps, specifically those described above. These simulations allow detailed exploration of the properties of the devices' number entry systems, and comparisons between different designs—there are many possible variations to experiment with, as described in more detail in the next section. In particular, several variants of the B.Braun Infusomat Space VTBI number entry interface have been implemented.

Workshop annotation mechanism — We also introduce the concept of state annotation as a research tool to enhance collaborative critique of an interactive system. Members of this session will be able to add annotations to any states in the interactive simulations to identify or mark issues regarding the usability, safety or design of the system being evaluated. Annotations will be automatically saved with information about the current state of the system as well as the user interactions that led to that state starting from power up, and are automatically shared among all clients connected to the simulation.

Commercial simulations — We have some commercial simulations, intended for hospital training purposes, including for versions C and D of the B.Braun; our physical pump is version E. The simulation diverges from observed behaviour (at least) in that it does not clamp to a minimum value as described above, but rather to 0. This suggests that the defaulting minimum value is introduced in version E.

Physical devices and manuals — Finally we will bring some real physical devices together with their operator manuals for comparison with our and the commercial simulations.

5. A TAXONOMY FOR NUMBER ENTRY

In order to support discussion in the workshop, in this section we propose an initial 'taxonomy' of features and behaviours of number entry interfaces, particularly considering some of the behaviours described above. We hope that further debate will refine and augment this list. As we describe the taxonomy we make some observations and speculations about relevance to usability, simplicity of conceptual models, etc. but our main purpose in this paper is to ask questions and so promote discussion, not provide answers specifically — most of this space remains intellectually unexplored.

At the top level, we distinguish between serial and incremental entry. Serial entry involves entering the number as a string, usually via a numeric keypad; consider entering a number into a desktop calculator, for example. Conversely, incremental entry involves making a series of incremental changes to some displayed value in order to obtain the desired value — often but not necessarily on a digit-by-digit basis. As incremental entry can be implemented using just a few keys, typically ▲ ▼ ◀ ▶, which may already be present for navigational purposes, it is a common style on the kinds of medical devices we are interested in. As such, and as it is used by each of our example devices, we concentrate on issues surrounding this style, though serial entry is still interesting and appropriate further exploration, are questions as to which style is preferable in general and in particular situations, and why.

Focusing on incremental number entry, we identify three major aspects of interest: basic behaviour (decade vs arithmetic, see figure 3); behaviour at minimum and maximum values (see figure 4); and digit visibility.

First we consider basic behaviour, which may be decade

or arithmetic style. In decade style, each digit is edited independently, and typically subject to wraparound at 0 and 9. For example, given a display of 1 9 2 . 4, if the user hits ▲, the new value is 1 0 2 . 4—the 9 increased by 1, modulo 10, wrapping round to 0, and all other digits are unaffected. In this style the number really must be dialled in one digit at a time.

In arithmetic style, user actions cause arithmetic modifications to the value displayed: add 1, subtract 10, etc. On the Alaris GP there are dedicated up/down buttons of differing magnitude; on the B.Braun ◄ and ► navigate between digits and ▲ and ▼ modify values. Repeating the previous example in arithmetic style leads to a display of 2 0 2 . 4 with the increment in the tens column being 'carried' to the hundreds. It is unclear if or when this would be preferable to users, though one can imagine that for fine adjustments around some value it is easier and would involve less ◄/► actions.

Either of these 'starting points' may be implemented using little code, and with very simple logic. (See our example simulations.) They each provide a clear conceptual model of the interface which users ought to be able to fathom completely with very little experimentation. Edge cases are often where problems arise; thus, what happens around the maximum and minimum values? There are a number of subtleties, not immediately obvious. First: what are the maximum and minimum values? Either might be a function of what we can fit in the display (which might change over time — see below), or some semantically-relevant value. The minimum could be the negative of the maximum, or (more often) zero, or something else. For VTBI entry on the B.Braun, the minimum is either 1 or 0.1 depending on digit visibility (see below), and can only be zeroed by an exact operation. Thus, for example, 0 1 . followed by ▼ leads to 0 0 . 1 ('minimum' value), whereas 1 . followed by ▼ leads to 0 . (true zero). This leads to some strange behaviour and a messy conceptual model, and we are presently unable to imagine any user-driven motivation for implementing this feature, though we note that 0 is not an allowed value for VTBI (the OK button doesn't work when the display is 0).

Assuming we know what the maximum and minimum ought to be, how should a device behave at those values? For the decade interface this issue can be ignored: the interface 'wraps round' naturally; one could in fact apply the following strategies in that context instead, but doing so breaks the conceptual model badly.

Arithmetic entry can also wrap round between min/max values, but now we are wrapping on the total value, not individual digits. Consider 0 0 0 on a display with boundaries at 0 and 9999, followed by ▼; this subtracts 100, taking us to 9 9 0 0. Then ▲ undoes this, adding 100 with wraparound, returning to

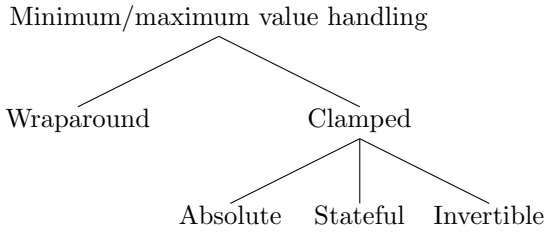

Figure 4. Number entry—boundary value handling

0 0 0. This retains a clean conceptual model, but with the danger of allowing large numbers to be easily entered accidentally: a single ▼ takes us from an initial (and safe) 0 to 9 9 9 9—though at least this is easy to undo.

More commonly, arithmetic interfaces restrict ('clamp') numbers to the boundaries. Here, we identify three approaches, which we call absolute clamping, stateful clamping, and invertible—see figure 4.

In *absolute clamping*, an attempt to move the value beyond a limit stops at the limit. E.g., 9 9 4 5 then ▲ leads to 9 9 9 9; similarly, 0 9 5 3 then ▼ leads to 0 0 0 0. This is a fairly natural behaviour, easy to program and conceptually clear once discovered; however, as it throws information away it could be annoying to users. In the face of annoyed users, a natural extension is *stateful clamping* where some state is introduced allowing accidental clamping operations to be undone. Here 9 9 4 5 then ▲ gives 9 9 9 9 but an immediate ▼ restores 9 9 4 5 (without state, we would get 9 8 9 9); anything other than ▼ throws away the state and disallows the undo. This is how VTBI entry on the B.Braun operates, for example.

In decade style ▲ and ▼ are inverses of each other, and it's always possible to undo the last change easily. This is lost with absolute clamping, even with state, e.g. 9 9 4 5 then ▲ ▲ ▼ ▼ gives 9 9 9 7 not 9 9 4 5. An extension which seeks to fix this without introducing wraparound is to make all successful operations *invertible*. Here, if an operation would take the value beyond its maximum or minimum, it doesn't happen, and this is indicated to the user via a beep (say). Now 9 9 4 5 then ▲ leaves the value unchanged, but the user is alerted that this is the case. The more general rule is: *any operation that does not have an inverse has no effect other than a warning such as a beep*; now the user knows, if they hear a beep, the normal inverse behaviour doesn't apply; otherwise, they know without looking that they can undo the last operation.

The third general area of interest we identify is that of digit visibility, around which there are several related issues. First, consider a decade-style system implemented in hardware — a physical device with one

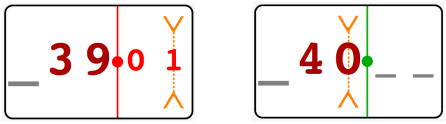

Figure 5. An improved number entry interface in action.

wheel per digit: spinning the wheel naturally wraps around modulo 10 (indeed, we obtain the name 'decade system' from such devices, which have one wheel per decade to be entered). On such a system, every digit is always visible, which can lead to confusion: for example it can be hard to distinguish 080.00 from 000.00. We are aware of two strategies for mitigating this: blanking leading/trailing zeros, and hiding digits entirely. The first strategy is obvious: only show significant digits. There are (at least) two questions to ask: what to display for blank (a space? an underscore?) and whether to 'follow the cursor' filling in zeros prospectively (e.g. do you display _0.01 or __.01?); the cognitive implications of either choice remain uninvestigated. On some systems we also see use of a second strategy, where digits are shown/hidden depending on the magnitude of the value being entered, usually on grounds of semantic relevance. For example and in particular, for VTBI mL entry, the B.Braun hides the hundredths and then tenths digits if the hundreds and thousands digits (respectively) are non-blank (including while 'following the cursor' as described above.) Similarly, ten-thousands is only shown if tenths is hidden. This is semantically sensible, but slightly disorienting to the user as the display is always right-aligned, so sometimes one digit disappears, another disappears, and the whole thing shifts to the right. Related to this: is the decimal point visible if no fractional digits are filled in? Canada's Institute for Safe Medication Practices (ISMP) says it should not be — and also mandates reducing the size of fractional digits, to more clearly distinguish 5.0 from 50 (say); changing colour may also be a worthwhile tactic here [3]. On the B.Braun, the decimal point is visible while the tenths column is visible, whether it is empty or not.

We've identified a large design space for the apparently simple question of incremental number entry; the task remains to identify the trade-offs each of these choices involves, and how they affect the conceptual mappings users build between their actions and their effects.

6. A SAMPLE BETTER INTERFACE?

Figure 5 shows a working mock-up of a potentially better user interface, to be operated by ◄ ► ▲ ▼ keys as on the B.Braun. It has several interesting features:

- The cursor (shown on the right-most digit position) and the decimal point are highly salient.
- Digits to the right of the decimal point are highlighted and smaller. The decimal point remains but is dimmed when the decimal digits are zero.
- Following good practice, leading and trailing zeroes are suppressed (shown as _). However, they behave *exactly* like 0 in how they are controlled by ▲ ▼.
- The number has upper and lower bounds (for the 5-digit example shown below, the bounds must be within 0 to 999.99).
- There is no hidden state. The behaviour of the interface is predictable from the display alone.
- Sometimes keys cannot work: as shown the ► cannot move the cursor further right; or if the display showed 999.99 no digit could be incremented; and so on. Whenever a key is pressed that cannot do anything, the interface beeps and otherwise does nothing. (Thus adding 1 to 999.00 does not increase it to the maximum value 999.99.)
- **Always**, a key beeps or its effect can be cancelled by pressing the opposite key: thus always ◄ ► and the other 3 pairs do nothing unless the first key pressed causes a beep, in which case the second key behaves normally.
- The rule above can be followed with the arithmetic style of interaction or with decade style. We prefer the arithmetic style, since after pressing ▲ or ▼ the number is *always* changed by $\pm 10^n$ or 0 if the key beeps. With the decade style, there can be a beep (if the number would hit a limit) or the number may change either by $\pm 10^n$ (most often) or at most $\pm 9 \times 10^n$ (about 1 in 10 times); this behaviour is much less predictable.
- Hence, ▲ ▼ work on arithmetic; that is, they always add $\pm 10^n$ to the displayed number (n depending solely on the cursor position), or they beep (and otherwise do nothing) if $\pm 10^n$ would have resulted in overflow.
- The design generalises readily, for instance to times by using different bases for each digit (i.e., base 10, 10, 6, 10 respectively, with an upper bound of 2359).
- If the application requires a movable decimal point, then ► pressed when the cursor is in the right-most column *and* the left-most digit is _ then the decimal point will move left (and conversely for ◄). This behaviour ensures no significant figures are ever lost and that the decimal point is always shown within the display. Again, the precision is limited by bounds and if the decimal point cannot move, then the key beeps.

Starting with the example on the left in figure 5: pressing ► (beeps and otherwise does nothing) then ▼ ◄ ◄ ▲ obtains the view on the left in figure 5. Notice number carry, moved cursor and changed decimal point style.

7. DISCUSSION AND FUTURE WORK

Our aim here is to start debate and exploration of these issues; future work is to continue that systematically. Here we identify some key challenges and opportunities.

A problem with work of this sort is that seemingly sensible design properties have unexpected impacts on how users behave. Therefore the workshop must help identify issues for empirically-based research. Consider, for example, the 'undo' design heuristic recommended by Nielsen [4]. How might we arrive at a more detailed set of properties for programmers of medical devices? Let us suppose we start by asking the following two research questions: 1) Is the 'undo' heuristic a significant affector for both serial and incremental number entry in terms of error rates? 2) Are error rates on systems in the same 'class' effected in similar ways by the level of undo offered? Formally-guided experimental investigation could help answer these questions. To avoid empirical experimentation on every possible variant of number entry, we might identify a set of distinct 'centroid-cases' (specific variants representative of some 'cluster' of similar variants), by preliminary exploration via a formal model of human-device interaction; this process could also produce a suitable feature-set for classifying different kinds of number entry system, formalising and completing the taxonomy suggested above. The results from experimental investigation following the formal modelling step would give a more precise description of the trends seen for these determining features of keypads with respect to 'undo' and error-rate.

The question of how users' mental models of number entry systems develop and relate to the developers' models and the code they write, is of particular interest. We propose that users of medical devices largely develop their mental models of device behaviour through interaction with the devices themselves, and by existing conventions; how can devices be designed to optimise this learning process, guiding users to the 'golden path'?

This paper has mainly described incremental interfaces; serial entry of course also needs to be explored. The related task of time entry is also critical and worthy of attention. For example, in the B.Braun most of the interface principles of the VTBI and Rate number entry interface are found in the time entry interface: pre-set maximums and minimums, jumping to the minimum if the edited number is less than the minimum, for example. Interestingly, if the VTBI is set to 99999 and we try to set the time, when we press ▲ on any position the time jumps up to 83:20; we remain unable to explain this behaviour.

8. CONCLUSIONS

Interactive number entry is deceptively complex, and particularly for dependable applications — medicine and healthcare — must be done well on the basis of a thorough analysis of requirements. This paper has therefore explored the related design issues and principles, and through case studies and analysis, developed potentially more dependable approaches. Ideally after appropriate empirical testing (particularly in real environments) and iterative design, this work will lead to a definitive approach for dependable number entry.

9. ACKNOWLEDGMENTS

Funded as part of the CHI+MED: Multidisciplinary Computer-Human Interaction research for the design and safe use of interactive medical devices project, EPSRC Grant Number EP/G059063/1 and Formally-based tools for user interface analysis and design, EPSRC Grant Number EP/F020031/1.

10. REFERENCES

1. D. P. Brumby, D. D. Salvucci, and A. Howes. Focus on driving: how cognitive constraints shape the adaptation of strategy when dialing while driving. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 1629–1638, New York, NY, USA, 2009. ACM.

2. R. Hourizi and P. Johnson. Unmasking mode errors: A new application of task knowledge principles to the knowledge gaps in cockpit design. In *Proceedings of Interact 2001, 8th IFIP TC Conf. on Human Computer Interaction*. IOS Press, 2001.

3. Institute for Safe Medication Practices. List of error-prone abbreviations, symbols and dose designations. www.ismp.org/tools/abbreviations, 2006.

4. J. Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

5. J. Reason. Human error: models and management. *BMJ*, 320(7237):768–770, March 2000.

6. H. Thimbleby and P. Cairns. Reducing number entry errors: Solving a widespread, serious problem. *Journal Royal Society Interface*, 7(51):1429–1439, 2010.

7. J. Zhang, V. L. Patel, T. R. Johnson, and E. H. Shortliffe. A cognitive taxonomy of medical errors. *J. of Biomedical Informatics*, 37:193–204, June 2004.